HIERARCHICAL DATA-DRIVEN SEARCH AND NAVIGATION SYSTEM AND METHOD FOR INFORMATION RETRIEVAL

This application is a continuation-in-part of App. Ser. No. 09/961,131, entitled "Scalable Hierarchical Navigation System and Method for Information Retrieval," filed September 21, 2001, which is a continuation-in-part of Application Ser. No. 09/573,305, entitled "Hierarchical Data-Driven Navigation System and Method for Information Retrieval," filed May 18, 2000, which are incorporated herein by this reference.

1. Field of the Invention

The present invention generally relates to information search and navigation systems.

2. Background of the Invention

Information retrieval from a database of information is an increasingly challenging problem, particularly on the World Wide Web (WWW), as increased computing power and networking infrastructure allow the aggregation of large amounts of information and widespread access to that information. A goal of the information retrieval process is to allow the identification of materials of interest to users.

As the number of materials that users may search and navigate increases, identifying relevant materials becomes increasingly important, but also increasingly difficult. Challenges posed by the information retrieval process include providing an intuitive, flexible user interface and completely and accurately identifying materials relevant to the user's needs within a reasonable amount of time. Another challenge is to provide an implementation of this user interface that is highly scalable, so that it can

readily be applied to the increasing amounts of information and demands to access that information. The information retrieval process comprehends two interrelated technical aspects, namely, information organization and access.

Current information search and navigation systems usually follow one of three paradigms. One type of information search and navigation system employs a database query system. In a typical database query system, a user formulates a structured query by specifying values for fixed data fields, and the system enumerates the documents whose data fields contain those values. PriceSCAN.com uses such an interface, for example. Generally, a database query system presents users with a form-based interface, converts the form input into a query in a formal database language, such as SQL, and then executes the query on a relational database management system. Disadvantages of typical query-based systems include that they allow users to make queries that return no documents and that they offer query modification options that lead only to further restriction of the result set (the documents that correspond to the user's specifications), rather than to expansion or extension of the result set. In addition, database query systems typically exhibit poor performance for large data sets or heavy access loads; they are often optimized for processing transactions rather than queries.

A second type of information search and navigation system is a free-text search engine. In a typical free-text search engine, the user enters an arbitrary text string, often in the form of a Boolean expression, and the system responds by enumerating the documents that contain matching text. Google.com, for example, includes a free-text search engine. Generally a free-text search engine presents users with a search form,

often a single line, and processes queries using a precomputed index. Generally this index associates each document with a large portion of the words contained in that document, without substantive consideration of the document's content. Accordingly, the result set is often a voluminous, disorganized list that mixes relevant and irrelevant documents. Although variations have been developed that attempt to determine the objective of the user's query and to provide relevance rankings to the result set or to otherwise narrow or organize the result set, these systems are limited and unreliable in achieving these objectives.

A third type of information search and navigation system is a tree-based directory. In a tree-based directory, the user generally starts at the root node of the tree and specifies a query by successively selecting refining branches that lead to other nodes in the tree. Shopping.yahoo.com uses a tree-based directory, for example. In a typical implementation, the hard-coded tree is stored in a data structure, and the same or another data structure maps documents to the node or nodes of the tree where they are located. A particular document is typically accessible from only one or, at most, a few, paths through the tree. The collection of navigation states is relatively static—while documents are commonly added to nodes in the directory, the structure of the directory typically remains the same. In a pure tree-based directory, the directory nodes are arranged such that there is a single root node from which all users start, and every other directory node can only be reached via a unique sequence of branches that the user selects from the root node. Such a directory imposes the limitation that the branches of the tree must be navigationally disjoint—even though the way that documents are assigned to the disjoint

branches may not be intuitive to users. It is possible to address this rigidity by adding additional links to convert the tree to a directed acyclic graph. Updating the directory structure remains a difficult task, and leaf nodes are especially prone to end up with large numbers of corresponding documents.

In all of these types of search and navigation systems, it may be difficult for a user to revise a query effectively after viewing its result set. In a database query system, users can add or remove terms from the query, but it is generally difficult for users to avoid underspecified queries (i.e. too many results) or overspecified queries (i.e. no results). The same problem arises in free-text search engines. In tree-based directories, the only means for users to revise a query is either to narrow it by selecting a branch or to generalize it by backing up to a previous branch.

Having an effective means of revising queries is useful in part because users often do not know exactly what they are looking for. Even users who do know what they are looking for may not be able to express their criteria precisely. And the state of the art in information retrieval technology cannot guarantee that even a precisely stated query will be interpreted as intended by the user. Indeed, it is unlikely that a perfect means for formation of a query even exists in theory. As a result, it is helpful that the information retrieval process be a dialogue with interactive responses between the user and the information retrieval system. This dialogue model may be more effectively implemented with an effective query revision process.

Some information retrieval systems combine a search engine with a vocabulary of words or phrases used to classify documents. These systems enable a three-step process

for information retrieval. In the first step, a user enters a text query into a search form, to which the system responds with a list of matching vocabulary terms. In the second step, the user selects from this list, to which the system responds with a list of documents. Finally, in the third step, the user selects a document.

A problem with such systems is that they typically do not consider the possibility that a user's search query may match a conjunction of two or more vocabulary terms, rather than an individual term. For example, in a system whose vocabulary consists of consumer electronics products and manufacturers, a search for *Sony DVD players* corresponds to a conjunction of two vocabulary terms: *Sony* and *DVD players*. Some systems may address this problem by expanding their vocabularies to include vocabulary terms that incorporate compound concepts (e.g., all valid combinations of manufacturers and products), but such an exhaustive approach is not practical when there are a large number of independent concepts in a system, such as product type, manufacturer, price, condition, etc. Such systems also may fail to return concise, usable search results, partially because the number of compound concepts becomes unmanageable. For example, a search for *software* in the Yahoo category directory returns 477 results, most of which represent compound concepts (e.g., *Health Care > Software*).

Various other systems for information retrieval are also available. For example. U.S. Patents Nos. 5,715,444 and 5,983,219 to Danish et al., both entitled "Method and System for Executing a Guided Parametric Search," disclose an interface for identifying a single item from a family of items. The interface provides users with a set of lists of features present in the family of items and identifies items that satisfy selected features.

Other search and navigation systems include i411's Discovery Engine, Cybrant's Information Engine, Mercado's IntuiFind, and Requisite Technology's BugsEye.

3. Summary of the Invention

The present invention, a highly scalable, hierarchical, data-driven information search and navigation system and method, enables the search and navigation of a collection of documents or other materials using certain common attributes associated with those materials. The search interface allows the user to enter queries that may correspond to either single terms or combinations of terms from a vocabulary used to classify documents. The navigation interface allows the user to select values for the attributes associated with the materials in the current navigation state and returns the materials that correspond to the user's selections. In some embodiments, the user's selections may be combined using Boolean operators. The present invention enables this navigation mode by associating terms (attribute-value pairs) with the documents, defining a set of hierarchical refinement relationships (i.e., a partial order) among the terms, and providing a guided navigation mechanism based on the association of terms with documents and the relationships among the terms.

The present invention includes several components and features relating to a hierarchical data-driven search and navigation system. Among these are a user interface, a knowledge base, a process for generating and maintaining the knowledge base, a navigable data structure and method for generating the data structure, WWW-based applications of the system, and methods of implementing the system. Although the invention is described herein primarily with reference to a WWW-based system for

navigating a product database, it should be understood that a similar search and navigation system could be employed in any database context where materials may be associated with terms and users can identify materials of interest by way of those terms.

The present invention uses a knowledge base of information regarding the collection of materials to formulate and to adapt the interface to guide the user through the collection of navigation states by providing relevant navigation options. The knowledge base includes an enumeration of attributes relevant to the materials, a range of values for each attribute, and a representation of the partial order that relates terms (the attribute-value pairs). Attribute-value pairs for materials relating to entertainment, for example, may be *Products*: *Movies* and *Director*: *Spike Lee*. (Attribute-value pairs are represented throughout this specification in this *Attribute: Value* format; navigation states are represented as bracketed expressions of attribute-value pairs.) The knowledge base also includes a classification mapping that associates each item in the collection of materials with a set of terms that characterize that item.

The knowledge base is typically organized by domains, which are sets of materials that conform to natural groupings. Preferably, a domain is chosen such that a manageable number of attributes suffice to effectively distinguish and to navigate among the materials in that domain. The knowledge base preferably includes a characterization of each domain, which might include rules or default expectations concerning the classification of documents in that domain. A particular item may be in more than one domain.

Embodiments of the present invention include a user interface for searching. This interface allows users to use a free-text search to find terms of interest. A free-text query may be composed of one or more words. The system may interpret free-text queries in various ways; the interpretations used to execute a free-text query will determine the nature of the search results for that query. A single-term interpretation maps the complete query to an individual term in the knowledge base. A multi-term interpretation maps the query to a conjunction of two or more terms in the knowledge base—that is, a plurality of terms that corresponds to a conjunctive navigation state. Depending on the particular implementation and application context, a free-text query may be mapped to one or more single-term interpretations, one or more multi-term interpretations, or a combination of both. In another aspect of the present invention, the user interface allows users to use a free-text search either merely to find matching terms or further to find navigation states or materials associated with the matching terms.

The present invention also includes a user interface for navigation. The user interface preferably presents the user's navigation state as an expression of terms organized by attribute. For a given expression of terms, the user interface presents materials that are associated with those terms in accordance with that expression and presents relevant navigation options for narrowing or for generalizing the navigation state. In one aspect of the present invention, users navigate through the collection of materials by selecting and deselecting terms.

In one aspect of the present invention, the user interface responds immediately to the selection or the deselection of terms, rather than waiting for the user to construct and

to submit a comprehensive query composed of multiple terms. Once a query has been executed, the user may narrow the navigation state by conjunctively selecting additional terms, or by refining existing terms. Alternatively, the user may broaden the navigation state by deselecting terms that have already been conjunctively selected or by generalizing the terms. In preferred embodiments, the user may broaden the navigation state by deselecting terms in an order different from that in which they were conjunctively selected. For example, a user could start at {*Products: Movies*}, narrow by conjunctively selecting an additional term to {*Products: Movies* AND *Genre: Drama*}, narrow again to {*Products: Movies* AND *Genre: Drama* AND *Director: Spike Lee*}, and then broaden by deselecting a term to {*Products: Movies* AND *Director: Spike Lee*}.

In another aspect of the present invention, the user may broaden the navigation state by disjunctively selecting additional terms. For example, a user could start at {*Products: DVDs*}, and then broaden by disjunctively selecting a term to {*Products: DVDs* OR *Products: Videos*}, and then narrow by conjunctively selecting a term to {(*Products: DVDs* OR *Products: Videos*) AND *Director: Spike Lee*}.

In another aspect of the present invention, the user may narrow the navigation state by negationally selecting additional terms. For example, a user could start at {*Products: DVDs*}, narrow by conjunctively selecting a term to {*Products: DVDs* AND *Genre: Comedy*}, and then narrow by negationally selecting a term to {*Products: DVDs* AND *Genre: Comedy* AND (NOT *Director: Woody Allen*)}.

In another aspect of the present invention, the user interface presents users with context-dependent navigation options for modifying the navigation state. The user

interface does not present the user with options whose selection would correspond to no documents in the resulting navigation state. Also, the user interface presents new navigation options as they become relevant. The knowledge base may contain rules that determine when particular attributes or terms are made available to users for navigation.

In another aspect of the invention—for example, when the materials correspond to products available for purchase from various sources—the knowledge base includes a catalog of canonical representations that have been aggregated from the materials.

In another aspect of the invention, the knowledge base may include definitions of stores, sets of materials that are grouped to be retrievable at one time. A store may include documents from one or more domains. An item may be assigned to more than one store. The knowledge base may also include rules to customize navigation for particular stores.

In another aspect of the invention, the knowledge base is developed through a multi-stage, iterative process. Workflow management allocates resources to maximize the efficiency of generating and of maintaining the knowledge base. The knowledge base is used to generate data structures that support navigation through a collection of materials. In one aspect of the invention, the system includes a hierarchy (i.e., a partial order) of navigation states that map expressions of terms to the sets of materials with which those terms are associated. In another aspect of the invention, the navigation states are related by transitions corresponding to terms used to narrow or broaden from one navigation state to another. The navigation states may be fully or partially precomputed, or may be entirely computed at run-time. In another aspect of the invention,

implementations of the invention may be scalable through parallel or distributed computation. In addition, implementations of the invention may employ master and slave servers arranged in a hierarchical configuration.

4. Brief Description of the Drawings

The invention, including these and other features thereof, may be more fully understood from the following description and accompanying drawings, in which:

Figure 1 is a view of a user interface to a search and navigation system in accordance with an embodiment of the present invention.

Figure 2 is a view of the user interface of Figure 1, showing a drop-down pick list of navigable terms.

Figure 3 is a view of the user interface of Figure 1, showing a navigation state.

Figure 4 is a view of the user interface of Figure 1, showing a navigation state.

Figure 5 is a view of the user interface of Figure 1, showing a navigation state.

Figure 6 is a view of the user interface of Figure 1, showing a navigation state.

Figure 7 is a view of the user interface of Figure 1, showing a navigation state.

Figure 8 is a view of the user interface of Figure 1, showing a navigation state.

Figure 9 is a view of the user interface of Figure 1, showing the result of a free-text search.

Figure 10 is a view of a user interface in accordance with another embodiment of the present invention, showing the result of a free-text search.

Figure 11 is a view of the user interface of Figure 10, showing the result of a free-text search.

Figure 12 is a view of a user interface in accordance with another embodiment of the invention, showing the result of a free-text search.

Figure 13 is a view of a user interface in accordance with another embodiment of the invention, showing information about a particular document.

Figures 14A-C are representative examples of how the range of values for an attribute could be partially ordered in accordance with an embodiment of the present invention.

Figure 15 is a block diagram of a process for collecting and classifying documents in accordance with an embodiment of the present invention.

Figure 16 is a table illustrating how a set of documents may be classified in accordance with an embodiment of the present invention.

Figure 17 is a representative partial order of navigation states in accordance with an embodiment of the present invention.

Figure 18 is a block diagram of a process for precomputing a navigation state in accordance with an embodiment of the present invention.

Figure 19 is a view of a user interface to a search and navigation system in accordance with an embodiment of the invention, showing disjunctive selection.

Figure 20 is a view of a user interface to a search and navigation system in accordance with an embodiment of the invention, showing disjunctive selection.

Figure 21 is a view of a user interface to a search and navigation system in accordance with an embodiment of the invention, showing negational selection.

Figure 22 is a view of a user interface to a search and navigation system in accordance with an embodiment of the invention, showing negational selection.

Figure 23 is a block diagram of a method for processing a free-text search query in accordance with an embodiment of the present invention.

Figure 24 is a block diagram of a system and a method for processing a request across multiple servers in accordance with an embodiment of the present invention.

Figure 25 is a flow diagram of steps for combining refinement options from slave servers in accordance with an embodiment of the present invention.

5. Detailed Description of the Preferred Embodiments

User Interface

In accordance with one embodiment of the present invention, Figure 1 shows a user interface 10 to a hierarchical, data-driven search and navigation system. The search and navigation system operates on a collection of documents defined in a knowledge base. As is shown, the user is preferably presented with at least two alternative methods of using the search and navigation system: (1) by selecting terms to navigate through the collection of documents, or (2) by entering a desired query in a search box.

The search and navigation system preferably organizes documents by domain. In accordance with one embodiment of the present invention, the user interface 10 shown in Figures 1-9 is operating on a set of documents that are part of a wine domain. Preferably, a domain defines a portion of the collection of documents that reflects a natural grouping. Generally, the set of attributes used to classify documents in a domain will be a manageable subset of the attributes used to classify the entire collection of documents. A

domain definition may be a type of product, e.g., wines or consumer electronics. A domain may be divided into subdomains to further organize the collection of documents. For example, there can be a consumer electronics domain that is divided into the subdomains of televisions, stereo equipment, etc. Documents may correspond to goods or services.

The user interface may allow users to navigate in one domain at a time. Alternatively, the user interface may allow the simultaneous navigation of multiple domains, particularly when certain attributes are common to multiple domains.

The user interface allows the user to navigate through a collection of navigation states. Each state is composed of an expression of terms and of the set of documents associated with those terms in accordance with that expression. In the embodiment shown in Figures 1-9, users navigate through the collection of navigation states by conjunctively selecting and deselecting terms to obtain the navigation state corresponding to each expression of conjunctively selected terms. Preferably, as in Figure 4, the user interface 10 presents a navigation state by displaying both the list 50 of terms 52 and a list 41 of some or all of the documents 42 that correspond to that state. Preferably, the user interface presents the terms 52 of the navigation state organized by attribute. Preferably, the initial navigation state is a root state that corresponds to no term selections and, therefore, to all of the documents in the collection.

As shown in Figure 2, the user interface 10 allows users to narrow the navigation state by choosing a value 28 for an attribute 22, or by replacing the currently selected value with a more specific one (if appropriate). Preferably, the user interface 10 presents

users with the options available to narrow the present navigation state, preferably with relevant terms organized by attribute. In some embodiments of the present invention, as shown in Figure 2, users can select values 28 from drop-down lists 26 denoted by indicators 24, that are organized by attributes 22 in the current navigation state. The user interface may present these navigation options in a variety of formats. For example, values can be presented as pictures or as symbols rather than as text. The interface may allow for any method of selecting terms, e.g., mouse clicks, keyboard strokes, or voice commands. The interface may be provided through various media and devices, such as television or WWW, and telephonic or wireless devices. Although discussed herein primarily as a visual interface, the interface may also include an audio component or be primarily audio-based.

Preferably, in the present navigation state, the user interface only presents options for narrowing the navigation state that lead to a navigation state with at least one document. This preferred criteria for providing navigation options ensures that there are no "dead ends," or navigation states that correspond to an empty result set.

Preferably, the user interface only presents options for narrowing the navigation state if they lead to a navigation state with strictly fewer documents than the present one. Doing so ensures that the user interface does not present the user with choices that are already implied by terms in the current navigation state.

Preferably, the user interface presents a new navigation state as soon as the user has chosen a term 28 to narrow the current navigation state, without any further

triggering action by the user. Because the system responds to each user with immediate feedback, the user need not formulate a comprehensive query and then submit the query.

In accordance with one embodiment of the present invention, as shown in Figures 3 and 4, the user interface 10 may enable broadening of the current navigation state by allowing the user to remove terms 52 from the list 50 of terms conjunctively selected. For example, the interface 10 may provide a list 50 with checkboxes 54 for removing selections and a button 56 to trigger the computation of the new navigation state. In the illustrated embodiment, the user can remove conjunctively selected terms 52 in any order and can remove more than one selection 52 at a time.

Preferably, the navigation options presented to the user are context-dependent. For example, terms that refine previously selected terms may become navigation options in the resulting navigation state. For example, referring to Figure 5, after the term *Flavors: Wood and Nut Flavors* 52 is conjunctively selected (the user has selected the value Wood and Nut Flavors 23 for the attribute Flavors), Wood and Nut Flavors 23 then appears in the interface for the new navigation state in the list 20 of attributes and allows conjunctive selection of values 28 that relate to that specific attribute for further refinement of the query. The user interface may also present certain attributes that were not presented initially, as they become newly relevant. For example, comparing Figure 3 to Figure 2, the attribute French Vineyards 25 appears in the list 20 of attributes only after the user has already conjunctively selected the term *Regions: French Regions* in a previous navigation state. Attributes may be embedded in this way to as many levels as are desired. Presenting attributes as navigation options when those attributes become

relevant avoids overwhelming the user with navigation options before those options are meaningful.

Additionally, for some attributes 22, multiple incomparable (non-refining) conjunctive selections of values 28 may be applicable. For example, for the attribute Flavor, the values Fruity and Nutty, neither of which refines the other, may both be conjunctively selected so that the terms *Flavors: Fruity* and *Flavors: Nutty* narrow the navigation state. Thus, users may sometimes be able to refine a query by conjunctively selecting multiple values under a single attribute.

Preferably, certain attributes will be eliminated as navigation options if they are no longer valid or helpful choices. For example, if all of the documents in the result set share a common term (in addition to the term(s) selected to reach the navigation state), then conjunctive selection of that term will not further refine the result set; thus, the attribute associated with that term is eliminated as a navigation option. For example, comparing Figure 6 with Figure 4, the attribute Wine Types 27 has been eliminated as a navigation option because all of the documents 42 in the result set share the same term, *Wine Types: Appellational Wines*. In preferred embodiments, an additional feature of the interface 10 is that this information is presented to the user as a common characteristic of the documents 42 in the result set. For example, referring to Figure 6, the interface 10 includes a display 60 that indicates the common characteristics of the documents 42 in the result set. Removing a term as a navigation option when all of the documents in the result set share that term prevents the user from wasting time by conjunctively selecting terms that do not refine the result set.

Preferably, the user interface also eliminates values as navigation options if their selection would result in no documents in the result set. For example, comparing Figure 8 to Figure 7, after the user selects the term *Wine Spectator Range: 95 - 100*, the user interface eliminates as navigation options all the values 28, 29 in the list 26 of values for the attribute Appellations 22 except for the values Alexander Valley 29 and Napa Valley 29. Alexander Valley 29 and Napa Valley 29 are the only two values in the list 26 of values for the attribute Appellations that return at least one document in the result set; all other values 28 return the empty set. Removing values as navigation options that would result in an empty result set saves the user time by preventing the user from reaching dead-ends.

Preferably, the user interface allows users to enter free-text search queries that may be composed of one or more words. The system may interpret free text queries in various ways. In particular, the system may map a free-text query to two types of search results: single-term interpretations and multi-term interpretations. A single-term interpretation maps the complete query to an individual term in the knowledge base. A multi-term interpretation maps the query to a conjunction of two or more terms in the knowledge base—that is, a plurality of terms that corresponds to a conjunctive navigation state. Depending on the particular implementation and application context, a free-text query may be mapped to one or more single-term interpretations, one or more multi-term interpretations, or a combination of both types of interpretations. In another aspect of the present invention, the user interface allows users to use a free-text search either to find matching terms or further to find materials associated with matching terms.

In accordance with one embodiment of the present invention, illustrated in Figure 9, in interface 90, a search box 30 preferably allows users to perform a free-text search for terms of interest, rather than performing a full-text search of the documents themselves. Preferably, the user interface responds to such a search by presenting a list 32 of single-term interpretations 33 including terms organized by attribute 36, and allowing the user to select from among them. Preferably, the user interface responds to the user's selection by presenting the user with the navigation state corresponding to the selection of that term. The user may then either navigate from that state (i.e., by narrowing or broadening it) or perform additional free-text searches for terms.

In accordance with another embodiment of the present invention, illustrated in Figure 10, the user interface 100 responds to free-text search queries by presenting a list 32 of multi-term interpretations 34, and allowing the user to select from among them. Preferably, the user interface responds to the user's selection by presenting the user with the navigation state corresponding to the selection of that conjunction of terms. The user may then either navigate from that state (i.e., by narrowing or broadening it) or perform additional free-text searches for terms.

In accordance with another embodiment of the present invention, illustrated in Figure 11, the user interface 100 responds to free-text search queries by presenting a list 32 of single-term interpretations 33 and multi-term interpretations 34, and allowing the user to select from among them. Preferably, the user interface responds to the user's selection by presenting the user with the navigation state corresponding to the selection

of that term or conjunction of terms. The user may then either navigate from that state (i.e., by narrowing or broadening it) or perform additional free-text searches for terms.

In accordance with another embodiment of the present invention, illustrated in Figure 12, the user interface 105 responds to free-text search queries by directly presenting the set of matching documents 35, for example, in accordance with full-text search of the documents. The user may then either navigate from that result (i.e., by narrowing or broadening it) or perform additional free-text searches for terms.

Preferably, the user interface 10 presents a full or partial list 41 of the documents that correspond to the current navigation state. Preferably, if a user is interested in a particular document 42, the user may select it and obtain a record 70 containing further information about it, including the list 72 of terms 74 that are associated with that document, as shown in Figure 13. Preferably, the user interface 10 allows the user to conjunctively select any subset of those terms 74 and thereby navigate to the navigation state that corresponds to the selected term expression.

Preferably, the user interface 10 also offers navigation options that directly link to an associated navigation state that is relevant to, but not necessarily a generalization or refinement of, the present navigation state. These links preferably infer the user's interests from the present navigation state and enable the user to cross-over to a related topic. For example, if the user is visiting a particular navigation state in a food domain, links may direct the user to navigation states of wines that would complement those foods in the wine domain.

In accordance with another embodiment of the present invention, the user is preferably presented with additional methods of using the search and navigation system such as: (1) by conjunctively selecting terms, (2) by disjunctively selecting terms, (3) by negationally selecting terms, or (4) by entering a desired keyword in a search box.

In another aspect of the present invention, the user may broaden the navigation state by disjunctively selecting additional terms. For example, a user could start at {*Products: DVDs*}, and then broaden by disjunctively selecting a term to {*Products: DVDs* OR *Products: Videos*}, and then narrow by conjunctively selecting a term to {(*Products: DVDs* OR *Products: Videos*) AND *Director: Spike Lee*}. Figure 19 shows a user interface 300 to a hierarchical, data-driven search and navigation system. The user interface 300 is operating on a collection of records relating to mutual funds. The interface 300 presents navigation options, including a list of attributes 310 relating to mutual funds and a list of terms 314 for a particular attribute 312, such as Fund Family, under consideration by a user. A selected term 316 is highlighted. As shown, the attribute-value pair {*Fund Family: Fidelity Investments*} has previously been selected. The illustrated search and navigation system allows the user to select attribute-value pairs disjunctively. As shown in Figure 20, after the user subsequently selects {*Fund Family: Vanguard Group*}in addition, the interface 300 presents a new navigation state {*Fund Family: Fidelity Investments* OR *Fund Family: Vanguard Group*}, including mutual funds 320 that match either selected attribute-value pair. Accordingly, both selected attribute-value pairs 316 are highlighted. In some embodiments, for example, to reduce computational requirements, disjunctive combination of attribute-value pairs may be

limited to mutually incomparable attribute-value pairs that correspond to the same attribute.

In another aspect of the present invention, the user may narrow the navigation state by negationally selecting additional terms. For example, a user could start at {*Products: DVDs*}, narrow by conjunctively selecting a term to {*Products: DVDs* AND *Genre: Comedy*}, and then narrow by negationally selecting a term to {*Products: DVDs* AND *Genre: Comedy* AND (NOT *Director: Woody Allen*)}. Figure 21 shows another interface 400 to a hierarchical, data-driven search and navigation system. The user interface 400 is operating on a collection of records relating to entertainment products. The user interface 400 includes a header 410 and a navigation area 412. The header 410 indicates the present navigation state {*Products: DVDs* AND *Genre:Drama*}, and implies the refinement options currently under consideration by the user. The leader "Not Directed By" 414 indicates a negational operation with respect to the Director attribute. The interface lists the attribute-value pairs 416 that can be combined with the expression for the present navigation state under this operation. As shown in Figure 22, after the user selects the term *Director: Martin Scorsese*, the interface 400 presents a new navigation state {*Products: DVDs* AND *Genre:Drama* AND (NOT *Director: Martin Scorsese*}.

Although the interface to the search and navigation system has been described herein as a user interface, the interface could provide other forms of access to the search and navigation system. In alternative embodiments, the interface may be an applications program interface to allow access to the search and navigation system for or through

other applications. The interface may also enhance the functionality of an independent data-oriented application. The interface may also be used in the context of a WWW-based application or an XML-based application. The search and navigation system may also support multiple interface modes simultaneously. The search and navigation system may be made available in a variety of ways, for example via wireless communications or on handheld devices.

Knowledge Base

Preferably, the search and navigation system stores all information relevant to navigation in a knowledge base. The knowledge base is the repository of information from two processes: taxonomy definition and classification. Taxonomy definition is the process of identifying the relevant attributes to characterize documents, determining the acceptable values for those attributes (such as a list or range of values), and defining a partial order of refinement relationships among terms (attribute-value pairs). Classification is the process of associating terms with documents. The knowledge base may also be used to maintain any information assets that support these two processes, such as domains, classification rules and default expectations. Additionally, the knowledge base may be used to maintain supplementary information and materials that affect users' navigation experience.

The taxonomy definition process identifies a set of attributes that appropriately characterize documents. A typical way to organize the taxonomy definition process is to arrange the collections of documents into domains, which are sets of documents that conform to a natural grouping and for which a manageable number of attributes suffice to

effectively distinguish and navigate among the documents in that domain. The knowledge base preferably includes a characterization of each domain, which might include rules or default expectations concerning the classification of documents in that domain.

The taxonomy definition process also identifies a full set of values, at varying levels of specificity when appropriate, for each attribute. The values preferably identify the specific properties of the documents in the collection. The values may be enumerated explicitly or defined implicitly. For example, for a "color" attribute, a full set of valid color values may be specified, but for a "price" or "date" attribute, a range within which the values may fall or a general data type, without defining a range, may be specified. The process of identifying these values may include researching the domain or analyzing the collection of documents.

The taxonomy definition process also defines a partial order of refinement relationships among terms (attribute-value pairs). For example, the term *Origin: France* could refine the term *Origin: Europe*. The refinement relationship is transitive and antisymmetric but not necessarily total. Transitivity means that, if term A refines term B and term B refines term C, then term A refines term C. For example, if *Origin: Paris* refines *Origin: France* and *Origin: France* refines *Origin: Europe*, then *Origin: Paris* refines *Origin: Europe*. Antisymmetry means that, if two terms are distinct, then both terms cannot refine each other. For example, if *Origin: Paris* refines *Origin: France*, then *Origin: France* does not refine *Origin: Paris*.

Further, the partial order of refinement relationships among terms is not necessarily a total one. For example, there could be two terms, *Origin: France* and

Origin: Spain, such that neither term refines the other. Two terms with this property are said to be incomparable. Generally, a set of two or more terms is mutually incomparable if, for every pair of distinct terms chosen from that set, the two terms are incomparable. Typically, but not necessarily, two terms with distinct attributes will be incomparable.

Given a set of terms, a term is a maximal term in that set if it does not refine any other terms in the set, and it is a minimal term in that set if no other term in the set refines it. For example, in the set {*Origin: France*, *Origin: Paris*, *Origin: Spain*, *Origin: Madrid*}, *Origin: France* and *Origin: Spain* are maximal, while *Origin: Paris* and *Origin: Madrid* are minimal. In the knowledge base, a term is a root term if it does not refine any other terms and a term is a leaf term if no other term refines it.

Figures 14A, 14B, and 14C illustrate attributes 112 and values 114, arranged in accordance with the partial order relationships, that could be used for classifying wines. The attributes 112 are Type/Varietal, Origin, and Vintage. Each attribute 112 corresponds to a maximal term for that attribute. An attribute 112 can have a flat set of mutually incomparable values (e.g., Vintage), a tree of values (e.g., Origin), or a general partial order that allows a value to refine a set of two or more mutually incomparable values (e.g., Type/Varietal). The arrows 113 indicate the refinement relationships among values 114.

Attributes and values may be identified and developed in several ways, including manual or automatic processing and the analysis of documents. Moreover, this kind of analysis may be top-down or bottom-up; that is, starting from root terms and working towards leaf terms, or starting from leaf terms and working towards root terms. Retailers,

or others who have an interest in using the present invention to disseminate information, may also define attributes and terms.

The classification process locates documents in the collection of navigation states by associating each document with a set of terms. Each document is associated with a set of mutually incomparable terms, e.g., {*Type/Varietal: Chianti*, *Origin: Italy*, *Vintage: 1996*}, as well as any other desired descriptive information. If a document is associated with a given term, then the document is also associated with all of the terms that the given term refines.

The classification process may proceed according to a variety of workflows. Documents may be classified in series or in parallel, and the automatic and manual classification steps may be performed one or more times and in any order. To improve accuracy and throughput, human experts may be assigned as specialists to oversee the classification task for particular subsets of the documents, or even particular attributes for particular subsets of the documents. In addition, the classification and taxonomy processes may be interleaved, especially as knowledge gained from one process allows improvements in the other.

Figure 15 illustrates the stages in a possible flow for the classification process 250. The data acquisition step 252, that is, the collection of documents for the database, may occur in several different ways. For example, a retailer with a product catalog over which the search and navigation system will operate might provide a set of documents describing its products as a pre-defined set. Alternatively, documents may be collected from one source, e.g., one Web site, or from a number of sources, e.g., multiple Web

sites, and then aggregated. If the desired documents are Web pages, the documents may be collected by appropriately crawling the Web, selecting documents, and discarding documents that do not fit in the domain. In the data translation step 254, the collected documents are formatted and parsed to facilitate further processing. In the automatic classification step 256, the formatted and parsed documents are processed in order to automatically associate documents with terms. In the manual classification step 258, human reviewers may verify and amend the automatic classifications, thereby ensuring quality control. Preferably, any rules or expectations violated in either the automatic classification step 256 or the manual classification step 258 would be flagged and presented to human reviewers as part of the manual classification step 258. If the collection of documents is divided into domains, then there will typically be rules that specify a certain minimal or preferred set of attributes used to classify documents from each domain, as well as other domain-specific classification rules. When the classification process is complete, each document will have a set of terms associated with it, which locate the document in the collection of navigation states.

In Figure 16, table 180 shows a possible representation of a collection of classified wine bottles. Preferably, each entry is associated with a document number 182, which could be a universal identifier, a name 184, and the associated terms 186. The name is preferably descriptive information that could allow the collection to be accessed via a free-text search engine as well as via term-based navigation.

In another aspect of the invention, the knowledge base also includes a catalog of canonical representations of documents. Each catalog entry represents a conceptually

distinct item that may be associated with one or more documents. The catalog allows aggregation of profile information from multiple documents that relate to the item, possibly from multiple sources. For example, if the same wine is sold by two vendors, and if one vendor provides vintage and geographic location information and another provides taste information, that information from the two vendors can be combined in the catalog entry for that type of wine. The catalog may also improve the efficiency of the classification process by eliminating duplicative profiling. In Figure 15, the catalog creation step 260 associates classified documents with catalog entries, creating new catalog entries when appropriate. For ease of reference, an item may be uniquely identified in the catalog by a universal identifier.

The knowledge base may also define stores, where a store is a subcollection of documents that are grouped to be retrievable at one time. For example, a particular online wine merchant may not wish to display documents corresponding to products sold by that merchant's competitors, even though the knowledge base may contain such documents. In this case, the knowledge base can define a store of documents that does not include wines sold by the merchant's competitors. In Figure 15, the store creation step 262 may define stores based on attributes, terms, or any other properties of documents. A document may be identified with more than one store. The knowledge base may also contain attributes or terms that have been customized for particular stores.

In Figure 15, the export process step 264 exports information from the knowledge base to another stage in the system that performs further processing necessary to generate a navigable data structure.

Navigation States

The search and navigation system represents, explicitly or implicitly, a collection of navigation states. A navigation state can be represented either by an expression of terms, or by the subset of the collection of documents that correspond to the term expression.

By way of example, types of navigation states include conjunctive navigation states, disjunctive navigation states and negational navigation states. Conjunctive navigation states are a special case of navigation states in which the term expression is conjunctive—that is, the expression combines terms using only the AND operator. Conjunctive navigation states are related by a partial order of refinement that is derived from the partial order that relates the terms.

In one aspect of the present invention, a conjunctive navigation state has two representations. First, a conjunctive navigation state corresponds to a subset of the collection of documents. Second, a conjunctive navigation state corresponds to a conjunctive expression of mutually incomparable terms. Figure 18 illustrates some navigation states for the documents and terms based on the wine example discussed above. For example, one navigation state 224 is {*Origin: South America*} (documents #1, #4, #5); a second navigation state 224 is {*Type/Varietal: White* AND *Origin: United States*} (documents #2, #9). The subset of documents corresponding to a conjunctive navigation state includes the documents that are commonly associated with all of the terms in the corresponding expression of mutually incomparable terms. At the same time, the expression of mutually incomparable terms corresponding to a conjunctive

navigation state includes all of the minimal terms from the terms that are common to the subset of documents, i.e., the terms that are commonly associated with every document in the subset. A conjunctive navigation state is preferably unique and fully specified; for a particular conjunctive expression of terms, or for a given set of documents, there is no more than one corresponding conjunctive navigation state.

One way preferred to define the collection of conjunctive navigation states is to uniquely identify each conjunctive navigation state by a canonical conjunctive expression of mutually incomparable terms. A two-step mapping process that maps an arbitrary conjunctive expression of terms to a canonical conjunctive expression of mutually incomparable terms creates states that satisfy this property. In the first step of the process, an arbitrary conjunctive expression of terms is mapped to the subset of documents that are associated with all of those terms. Recalling that if a document is associated with a given term, then the document is also associated with all of the terms that the given term refines, in the second step of the process, this subset of documents is mapped to the conjunctive expression of minimal terms among the terms that are common to all of the documents in that document set. The result of this second step is a conjunctive expression of mutually incomparable terms that uniquely identifies the corresponding subset of documents, and, hence, is a canonical representation for a conjunctive navigation state. By way of illustration, referring to the wine example in Figure 17, the term expression {*Origin: France*} maps to the subset of documents {documents #8, #11}, which in turn maps to the canonical term expression {*Type/Varietal: Red* AND *Origin: France*}.

The conjunctive navigation states 222, 224, 226 are related by a partial order of refinement relationships 220 derived from the partial order that relates terms. This partial order can be expressed in terms of either the subsets of documents or the term expressions that define a conjunctive navigation state. Expressed in terms of subsets of documents, a navigation state A refines a navigation state B if the set of documents that corresponds to state A is a subset of the set of documents that corresponds to state B. Expressed in terms of term expressions, a conjunctive navigation state A refines a conjunctive navigation state B if all of the terms in state B either are in state A or are refined by terms in state A. Referring to Figure 17, the navigation state 226 corresponding to the term expression {*Type/Varietal: Red* AND *Origin: Chile*} (document #4) refines the navigation state 224 corresponding to {*Origin: Chile*} (documents #4, #5). Since the refinement relationships among navigation states give rise to a partial order, they are transitive and antisymmetric. In the example, {*Type/Varietal: Red* AND *Origin: Chile*} (document #4) refines {*Origin: Chile*} (documents #4, #5) and {*Origin: Chile*} (documents #4, #5) refines {*Origin: South America*} (documents #1, #4, #5); therefore, {*Type/Varietal: Red* AND *Origin: Chile*} (document #4) refines {*Origin: South America*} (documents #1, #4, #5). The root navigation state 222 is defined to be the navigation state corresponding to the entire collection of documents. The leaf navigation states 226 are defined to be those that cannot be further refined, and often (though not necessarily) correspond to individual documents. There can be arbitrarily many intermediate navigation states 224 between the root 222 and the leaves 226. Given a pair of navigation states A and B where B refines A, there can be multiple paths of

intermediate navigation states 224 connecting A to B in the partial order. For convenience of definition in reference to the implementation described herein, a navigation state is considered to refine itself.

A user browses the collection of documents by visiting a sequence of one or more navigation states typically starting at the root navigation state 222. In one embodiment of the present invention, there are three basic modes of navigation among these states. The first mode is refinement, or moving from the current navigation state to a navigation state that refines it. The user can perform refinement either by adding a term through conjunctive selection to the current navigation state or by refining a term in the current navigation state; i.e., replacing a term with a refinement of that term. After the user adds or refines a term, the new term expression can be mapped to a canonical term expression according to the two-step mapping described above. The second mode is generalization, or moving from the current navigation state to a more general navigation state that the current state refines. The user can perform generalization either by removing a term from the current navigation state or by generalizing a term in the current navigation state; i.e., replacing a current term with a term that the current term refines. After the user removes or generalizes a term, the new term expression can be mapped to a canonical term expression. The third mode is simply creating a query in the form of a desired term expression, which again can be mapped to a canonical term expression to obtain a navigation state.

In other embodiments of the present invention, there are additional modes of navigation. In systems that support the corresponding types of navigation states, these

modes may include generalization of the navigation state through disjunctive selection, as shown in Figure 19, as well as refinement of the navigation state through negational selection, as shown in Figure 20. In general, terms can be combined using Boolean logic. Although term expressions that are not conjunctive do not necessarily have canonical forms, some implementations may be based on a system that uses a collection of conjunctive navigation states. One implementation is based on logical equivalence rules as described below.

Implementation

The knowledge base is transferred to a navigable data structure in order to implement the present invention. The navigation states may be fully precomputed, computed dynamically at run-time, or partially precomputed. A cache may be used to avoid redundant computation of navigation states.

In preferred embodiments, the collection of conjunctive navigation states may be represented as a graph—preferably, a directed acyclic multigraph with labeled edges. A graph is a combinatorial structure consisting of nodes and edges, where each edge links a pair of nodes. The two nodes linked by an edge are called its endpoints. With respect to the present invention, the nodes correspond to conjunctive navigation states, and the edges represent transitions that refine from one conjunctive navigation state to another. Since refinement is directional, each edge is directed from the more general node to the node that refines it. Because there is a partial order on the navigation states, there can be no directed cycles in the graph, i.e., the graph is acyclic. Preferably, the graph is a multigraph, since it allows the possibility of multiple edges connecting a given pair of

nodes. Each edge is labeled with a term. Each edge has the property that starting with the term set of the more general end point, adding the edge term, and using the two-step map to put this term set into canonical form leads to a refinement which results in the navigation state that is the other endpoint. That is, each edge represents a refinement transition between nodes based on the addition of a single term.

The following definitions are useful for understanding the structure of the graph: descendant, ancestor, least common ancestor (LCA), proper ancestor, proper descendant, and greatest lower bound (GLB). These definitions apply to the refinement partial order among terms and among nodes. If A and B are terms and B refines A, then B is said to be a descendant of A and A is said to be an ancestor of B. If, furthermore, A and B are distinct terms, then B is said to be a proper descendant of A and A is said to be a proper ancestor of B. The same definitions apply if A and B are both nodes.

If C is an ancestor of A and C is also an ancestor of B, then C is said to be a common ancestor of A and B, where A, B, and C are either all terms or all nodes. The minimal elements of the set of common ancestors of A and B are called the least common ancestors (LCAs) of A and B. If no term has a pair of incomparable ancestors, then the LCA of two terms—or of two nodes—is unique. For example, the LCA of *Origin: Argentina* and *Origin: Chile* is *Origin: South America* in the partial order of terms 110 of Figure 14B. In general, however, there may be a set of LCAs for a given pair of terms or nodes.

In an implementation that fully precomputes the collection of nodes, computation of the nodes in the graphs is preferably performed bottom-up.

The leaf nodes in the graph—that is, the nodes corresponding to leaf navigation states—may be computed directly from the classified documents. Typically, but not necessarily, a leaf node will correspond to a set containing a single document. The remaining, non-leaf nodes are obtained by computing the LCA-closure of the leaf nodes—that is, all of the nodes that are the LCAs of subsets of the leaf nodes.

The edges of the graph are determined according to a refinement function, called the R function for notational convenience. The R function takes as arguments two nodes A and B, where A is a proper ancestor of B, and returns the set of maximal terms such that, if term C is in R (A, B), then refining node A with term C results in a node that is a proper descendant of A and an ancestor (not necessarily proper) of B. For example, in Figure 17, R ({*Type/Varietal: Red*}, {*Type/Varietal: Merlot* AND *Origin: Argentina* AND *Vintage: 1998*}) = {*Type/Varietal: Merlot* AND *Origin: South America* AND *Vintage: 1998*}. If B_1 is an ancestor of B_2, then R (A, B_1) is a subset of R (A, B_2)—assuming that A is a proper ancestor of both B_1 and B_2. For example, R ({*Type/Varietal: Red*}, {*Type/Varietal: Red* AND *Origin: South America*}) = {*Origin: South America*}.

In the graph, the edges between nodes A and B will correspond to a subset of the terms in R (A, B). Also, no two edges from a single ancestor node A use the same term for refinement. If node A has a collection of descendant nodes {B_1, B_2,…} such that term C is in all of the R (A, B_i), then the only edge from node A with term C goes to LCA (B_1, B_2,…), which is guaranteed to be the unique maximal node among the B_i. In Figure 17, for example, the edge from node {*Type/Varietal: Red*} with term *Origin: South America* goes to node {*Type/Varietal: Red* AND *Origin: South America*} rather

than to that node's proper descendants {*Type/Varietal: Merlot* AND *Origin: South America* AND *Vintage: 1998*} and {*Type/Varietal: Red* AND *Origin: Chile*}. The LCA-closure property of the graph ensures the existence of a unique maximal node among the B_i. Thus, each edge maps a node-term pair uniquely to a proper descendant of that node.

The LCA-closure of the graph results in the useful property that, for a given term set S, the set of nodes whose term sets refine S has a unique maximal node. This node is called the greatest lower bound (GLB) of S.

The graph may be computed explicitly and stored in a combinatorial data structure; it may be represented implicitly in a structure that does not necessarily contain explicit representations of the nodes and edges; or it may be represented using a method that combines these strategies. Because the search and navigation system will typically operate on a large collection of documents, it is preferred that the graph be represented by a method that is scalable.

The graph could be obtained by computing the LCAs of every possible subset of leaf nodes. Such an approach, however, grows exponentially in the number of leaf nodes, and is inherently not scalable. An alternative strategy for obtaining the LCA closure is to repeatedly consider all pairs of nodes in the graph, check if each pair's LCA is in the graph, and add that LCA to the graph as needed. This strategy, though a significant improvement on the previous one, is still relatively not scalable.

A more efficient way to precompute the nodes is to process the document set sequentially, compute the node for each document, and add that node to the graph along with any other nodes necessary to maintain LCA-closure. The system stores the nodes

and edges as a directed acyclic multigraph. The graph is initialized to contain a single node corresponding to the empty term set, the root node. Referring to Figure 18, in process 230 for inserting a new node into the graph, in step 232, for each new document to be inserted into the graph that does not correspond to an existing node, the system creates a new node. In step 234, before inserting the new node into the graph, the system recursively generates and inserts any missing LCA nodes between the root node (or ancestor node) and the new node. To ensure LCA-closure after every node insertion, the system inserts the document node last, in steps 236 and 238, after inserting all the other nodes that are proper ancestors of it.

Inserting a new node requires the addition of the appropriate edges from ancestors to the node, in step 236, and to descendants out of the new node, in step 238. The edges into the node are preferably determined by identifying the ancestors that have refinement terms that lead into the new node and do not already have those refinement terms used on edges leading to intermediate ancestors of the new node. The edges out of the node are preferably determined by computing the GLB of the new node and appropriately adding edges from the new node to the GLB and to nodes to which the GLB has edges.

The entire graph of conjunctive navigation states may be precomputed by following the above procedures for each document in the collection. Computation of other types of navigation states is discussed below. Precomputing of the graph may be preferred where the size of the graph is manageable, or if users are likely to visit every navigation state with equal probability. In practice, however, users typically visit some navigation states more frequently than others. Indeed, as the graph gets larger, some

navigation states may never be visited at all. Unfortunately, reliable predictions of the frequency with which navigation states will be visited are difficult. In addition, it is generally not practical to precompute the collection of navigation states that are not conjunctive, as this collection is usually much larger than the collection of conjunctive navigation states.

An alternative strategy to precomputing the navigation states is to create indexes that allow the navigation states to be computed dynamically. Specifically, each document can be indexed by all of the terms that are associated with that document or that have refinements associated with that document. The resulting index is generally much smaller in size than a data structure that stores the graph of navigation states. This dynamic approach may save space and precomputation time, but it may do so at the cost of higher response times or greater computational requirements for operation. A dynamic implementation may use a one-argument version of the R function that returns all refinement terms from a given navigation state, as well a procedure for computing the GLB of a term set.

It is also possible to precompute a subset of the navigation states. It is preferable to precompute the states that will cost the most to compute dynamically. For example, if a state corresponds to a large subset of the documents, it may be preferable to compute it in advance. In one possible partial precomputation approach, all navigation states, particularly conjunctive ones, corresponding to a subset of documents above a threshold size may be precomputed. Precomputing a state is also preferable if the state will be visited frequently. In some instances it may be possible to predict the frequency with

which a navigation state will be visited. Even if the frequency with which a navigation state will be visited cannot be predicted in advance, the need to continually recompute can be reduced by caching the results of dynamic computation. Most recently or most frequently visited states may be cached.

As described above with respect to the interface, the system supports at least three kinds of navigational operations—namely refinement, generalization, and query by specifying an expression of terms. These operations may be further described in terms of the graph. For query refinement, the system enumerates the terms that are on edges from the node corresponding to the current navigation state. When the user selects a term for refinement, the system responds by presenting the node to which that edge leads. Similarly, for query generalization options, the system enumerates and selects edges that lead to (rather than from) the node corresponding to the current navigation state. Alternatively, query generalization may be implemented as a special case of query by specifying a set of terms. For query by specifying a set of keywords, the system creates a virtual node corresponding to the given term set and determines the GLB of the virtual node in the graph. If no GLB is found, then there are no documents that satisfy the query. Otherwise, the GLB node will be the most general node in the graph that corresponds to a navigation state where all documents satisfy the query.

The above discussion focuses on how the system represents and computes conjunctive navigation states. In some embodiments of the present invention, the user interface only allows users to navigate among the collection of conjunctive navigation states. In other embodiments, however, users can navigate to navigation states that are

not conjunctive. In particular, when the system supports navigation states that are not conjunctive, the user interface may allow users to select terms disjunctively or negationally.

If the system includes navigation states that are both conjunctive and disjunctive (e.g., {(*Products: DVDs* OR *Products: Videos*) AND *Director: Spike Lee*}), then in some embodiments, the system only precomputes a subset of the states, particularly if the total number of navigation states is likely to be too large to maintain in memory or even secondary (e.g., disk) storage. By using rules for equivalence of Boolean expressions, it is possible to express any navigation state that mixes conjunction and disjunction in terms of a union of conjunctive navigation states. The above example can be rewritten as {(*Products: DVDs* AND *Director: Spike Lee*) OR (*Products: Videos* AND *Director: Spike Lee*)}. This approach leads to an implementation combining conjunctive and disjunctive navigation states based on the above discussion, regardless of whether all, some, or none of the graph of conjunctive navigation states is precomputed.

In preferred embodiments, disjunctive selections may be made within, but not between, attributes. When determining the set of disjunctive generalizations, the system does not consider other terms from the attribute of the given disjunction to be in the navigation state. For example, if the navigation state is {*Type/Varietal: Red* AND *Origin: Chile*} and the system is allowing the disjunctive selection of other countries of origin, then the GLB and R function will be applied to the set {*Type/Varietal: Red*} rather than to {*Type/Varietal: Red* AND *Origin: Chile*}. Accordingly, the other terms for

the attribute of country of origin that are incomparable to "Chile" become generalization options for the navigation state.

If the system includes navigation states that use negation (e.g., {*Products: DVDs* AND *Genre: Comedy* AND (NOT *Director: Woody Allen*)}), then the negationally selected terms can be applied to navigation states as a post-process filtering operation. The above example can be implemented by taking the conjunctive navigation state {*Products: DVDs* AND *Genre: Comedy*} and applying a filter to it that excludes all movies associated with the term *Director: Woody Allen*. This approach leads to an implementation including negational navigation states based on the above discussion, regardless of whether all, some, or none of the graph of conjunctive navigation states is precomputed.

As with disjunction, when determining the set of negational generalizations, the system does not consider other terms from the attribute of the given negation to be in the navigation state. For example, if the navigation state is {*Medium: Compact Disc* AND *Artist: Prince*} and the system is allowing the negational selection of other artists (e.g., {*Artist: Prince* AND NOT (*Artist: The Revolution*)}), then the GLB and R function will be applied to the set {*Medium: Compact Disc*} rather than to {*Medium: Compact Disc* AND *Artist: Prince*}.

Another aspect of the present invention is the interpretation of free-text search queries. As discussed above, in embodiments of the present invention, a free-text query may be interpreted in two ways. A single-term interpretation maps the query to an individual term in the knowledge base. A multi-term interpretation maps the query to a

conjunction of two or more terms in the knowledge base—that is, a combination of terms that corresponds to a conjunctive navigation state.

A free-text query may be formed of one or more words. In a preferred embodiment of the invention, a single-term interpretation of a free-text query maps the query to a term that either contains or is associated with a word or words in that query. A query may have more than one single-term interpretation. For example, a query of *computer science* might have {*Department: Computer Science Department*} and {*School: School of Computer Science* } as single-term interpretations. For another example, a query of *zinfandel* might have {*Wine Type: Zinfandel*} and { *Wine Type: White Zinfandel*} as single-term interpretations. Various query semantics can be used to parse the search query and determine the set of single-term interpretations for a given free-text query. Under conjunctive query semantics, a matching term must contain all of the words in the query. Under disjunctive query semantics, a matching term must contain at least one of the words in the query. Under partial match query semantics, a matching term must contain a subset of the words in the query; the particular rules are application-dependent. It is also possible to vary the above to meet particular application needs. Variations include ignoring common "stop words" such as *the* and *of*, treating related words or word forms (e.g., singular and plural forms of nouns, or synonyms) as equivalent, automatic spelling correction, allowing delimited phrases (i.e., two or more words required to occur contiguously in a phrase), and support for negation (i.e., excluding terms that contain particular words).

In a preferred embodiment of the invention, a multi-term interpretation of a free-text query maps the query to a conjunction of terms that either contain or are associated with a word or words in that query and that correspond to a conjunctive navigation state in the system. A query may have more than one multi-term interpretation. For example, a query of *security books* might have {*Media Type: Books* AND *Subject: Computer Security*} and {*Media Type: Books* AND *Subject: Financial Security*} as multi-term interpretations. As with single-term interpretations, various query semantics can be used to parse the query and determine the set of multi-term interpretations for a given free-text query. Under conjunctive query semantics, a matching conjunction of terms must contain all of the words in the query. Under partial match query semantics, a matching conjunction of terms must contain a subset of the words in the query; the particular rules are application-dependent. It is also possible to vary the above to meet particular application needs. Variations, as discussed above, include ignoring common "stop words", treating related words or word forms as equivalent, automatic spelling correction, allowing delimited phrases, and support for negation. Regardless of the query semantics used, multi-term interpretations are themselves conjunctions of terms, and therefore preferably correspond to conjunctive navigation states.

In typical embodiments, one-word queries will only have single-term interpretations, while multi-word queries may have single-term interpretations, multi-term interpretations, or both. For example, a query of *casual shoes* might have {*Type: Casual Shoes*} as a single-term interpretation and {*Type: Athletic Shoes* AND *Merchant: Casual Living*} as a multi-term interpretation.

In a preferred embodiment of the invention, a multi-term interpretation is minimal—that is, the removal of any term from the interpretation would result in an interpretation that no longer satisfies the query. For example, the conjunction of terms {*Media Type: Books* AND *Subject: Computer Security*} is a minimal interpretation of the query *security books*; it is not, however, a minimal interpretation of the query *computer security*, since removing the term {*Media Type: Books*} results in the single-term interpretation {*Subject: Computer Security*} that satisfies the query. For another example, the conjunction of terms {*Flower Type: Red Roses* AND *Quantity: Dozen*} is a minimal interpretation of the query *dozen red roses*; in contrast, the conjunction of terms {*Flower Type: Red Roses* AND *Quantity: Dozen* AND *Color: Red*}is not a minimal interpretation for this query, since removing the term {*Color: Red*} results is a minimal multi-term interpretation that satisfies the query. In a preferred embodiment of the invention, disjunctive query semantics are not used for multi-term interpretations. Under disjunctive query semantics, all minimal interpretations of a query are single-term interpretations. Single-term interpretations are always minimal, since there is only one term to remove.

In a preferred embodiment of the invention, the computation of single-term search results uses an inverted index data structure that maps words to the terms containing them. Conjunctive query semantics may be implemented by computing the intersection of the term sets returned by looking up each query word in the inverted index, while disjunctive query semantics may be implemented by computing the union of the term sets.

In a preferred embodiment of the invention, the computation of multi-term search results uses both an inverted index data structure that maps words to the terms containing them and an inverted index data structure that maps terms to the materials associated with them.

In a preferred embodiment of the invention, the computation of multi-term search results that correspond to conjunctive navigation states involves a four step procedure. However, alternative procedures may be used. The steps of an algorithm 600 for receiving a query and returning the multi-term search results are indicated in Figure 23.

Once a query is received in step 610, in the first step 620, the system determines the set of terms that contain at least one word in the search query. This step is equivalent to computing single-term search results using disjunctive query semantics.

In the second step 630, the system partitions the set of terms into equivalence classes. Each equivalence class corresponds to a non-empty subset of words from the query. Two terms which match the same subset of words in the query will be assigned to the same equivalence class.

In the third step 640, the system considers the word subsets corresponding to the equivalence classes from the previous step, and computes which combinations of these subsets correspond to minimal interpretations of the query. A combination of subsets corresponds to a minimal interpretation if its union is equal to the entire query, but the removal of any of its subsets causes the union to not be equal to the entire query.

In the fourth step 650, the system considers, for each combination of subsets that corresponds to a minimal interpretation, the set of multi-term interpretations that can be

obtained from terms in the corresponding equivalence classes. These multi-term interpretations can be computed by enumerating, for each combination of subsets, all possible ways of choosing one term from each of the equivalence classes in the combination. Each resulting set of terms that corresponds to a conjunctive navigation state is added to the set of search results as a single-term (if it only contains one term) or multi-term interpretation. Finally in step 660, the results are returned.

For example, a search query of *1996 sweet red* in the wines domain obtains multi-term interpretations as follows.

In the first step, the following terms contain at least one of the words in the query:

Year: 1996

Wine Types: Sweet Wines

Flavors: Sweet

Wine Types: Appellational Red

Wine Types: Red Wines

Wineries: Red Birch

Wineries: Red Hill

In the second step, there are 3 equivalence classes:

Terms containing *1996*

Year: 1996

Terms containing *sweet*

Wine Types: Sweet Wines

Flavors: Sweet

Terms containing *red*

Wine Types: Appellational Red

Wine Types: Red Wines

Wineries: Red Birch

Wineries: Red Hill

In the third step, there is 1 combination of equivalence classes that is a minimal interpretation—namely, the combination of all 3 equivalence classes.

In the fourth step, the 8 candidates for minimal interpretations are:

{*Year: 1996* AND *Wine Types: Sweet Wines* AND *Wine Types: Appellational Red*}

{*Year: 1996* AND *Wine Types: Sweet Wines* AND *Wine Types: Red Wines*}

{*Year: 1996* AND *Wine Types: Sweet Wines* AND *Wineries: Red Birch*}

{*Year: 1996* AND *Wine Types: Sweet Wines* AND *Wineries: Red Hill*}

{*Year: 1996* AND *Flavors: Sweet* AND *Wine Types: Appellational Red*}

{*Year: 1996* AND *Flavors: Sweet* AND *Wine Types: Red Wines*}

{*Year: 1996* AND *Flavors: Sweet* AND *Wineries: Red Birch*}

{*Year: 1996* AND *Flavors: Sweet* AND *Wineries: Red Hill*}

Of these, the following map to conjunctive navigation states in the system and are thus returned as search results:

{*Year: 1996* AND *Wine Types: Sweet Wines* AND *Wineries: Red Birch*}

{*Year: 1996* AND *Wine Types: Sweet Wines* AND *Wineries: Red Hill*}

{*Year: 1996* AND *Flavors: Sweet* AND *Wine Types: Appellational Red*}

{*Year: 1996* AND *Flavors: Sweet* AND *Wine Types: Red Wines*}

{*Year: 1996* AND *Flavors: Sweet* AND *Wineries: Red Birch*}

{*Year: 1996* AND *Flavors: Sweet* AND *Wineries: Red Hill*}

The other two minimal interpretations do not having matching documents and do not map to a navigation state in the system.

For another example, a search query of *casual shoes* obtains multi-term interpretations as follows.

In the first step, the following terms contain at least one of the words in the query:

Type: Casual Shoes

Merchant: Casual Living

Brand: Casual Workstyles

Type: Athletic Shoes

Type: Dress Shoes

Brand: Goody Two Shoes

Merchant: Simple Shoes

In the second step, there are 3 equivalence classes:

Terms containing *casual*

Merchant: Casual Living

Brand: Casual Workstyles

Terms containing *shoes*

Type: Athletic Shoes

Type: Dress Shoes

Brand: Goody Two Shoes

Merchant: Simple Shoes

Terms containing both *casual* and *shoes*

Type: Casual Shoes

In the third step, there are 2 combinations of equivalence classes that are minimal interpretations. The first combination consists of the first two equivalence classes. The second combination consists of the third equivalence class by itself.

In the fourth step, the 9 candidates for minimal interpretations are:

{*Merchant: Casual Living* AND *Type: Athletic Shoes*}

{*Merchant: Casual Living* AND *Type: Dress Shoes*}

{*Merchant: Casual Living* AND *Brand: Goody Two Shoes*}

{*Merchant: Casual Living* AND *Merchant: Simple Shoes*}

{*Brand: Casual Workstyles* AND *Type: Athletic Shoes*}

{*Brand: Casual Workstyles* AND *Type: Dress Shoes*}

{*Brand: Casual Workstyles* AND *Brand: Goody Two Shoes*}

{*Brand: Casual Workstyles* AND *Merchant: Simple Shoes*}

{*Type: Casual Shoes*}

Of these, the following map to conjunctive navigation states in the system and are thus returned as search results:

{*Merchant: Casual Living* AND *Type: Athletic Shoes*}

{*Type: Casual Shoes*}

The other minimal interpretations do not have matching documents and do not map to a navigation state in the system. For example, the brand Casual Workstyles does not sell Athletic Shoes in the system.

Another aspect of the present invention is its scalability through parallel or distributed computation. One way to define scalability in a search and navigation system is in terms of four problem dimensions: the number of materials in the collection, the number of terms associated with each material in the collection, the rate at which the system processes queries (throughput), and the time necessary to process a query (latency). In this definition, a system as scalable if it can be scaled along any of these four dimensions at a subquadratic cost. In other words:

1. If the number of materials in the collection is denoted by the variable $\mathbf{n_1}$ and the other three problem dimensions are held constant, then the resource requirements are subquadratic in $\mathbf{n_1}$.

2. If the number of terms associated with each material in the collection is denoted by the variable $\mathbf{n_2}$ and the other three problem dimensions are held constant, then the resource requirements are subquadratic in $\mathbf{n_2}$.

3. If the number of queries that the system processes per second (i.e., the throughput) is denoted by the variable $\mathbf{n_3}$ and the other three problem dimensions are held constant, then the resource requirements are subquadratic in $\mathbf{n_3}$.

4. If the time necessary to process a query (i.e., the latency) is denoted by the variable $\mathbf{n_4}$ and the other three problem dimensions are held constant, then the resource requirements are subquadratic in $\mathbf{1/n_4}$.

Preferably, these resource requirements would be not only subquadratic, but linear. Also included within the concept of scalability, there is an allowance for overhead in creating a network of distributed resources. Typically, this overhead will be logarithmic, since the resources may be arranged in a hierarchical configuration of bounded fan-out.

In some embodiments, the present invention surmounts the limitations of a single computational server's limited resources by allowing for distributing the task of

computing the information associated with a navigation state onto a hierarchy of multiple computational servers that act in parallel.

One insight that drives this aspect of the present invention is that it is possible to partition the collection of materials among multiple "slave" servers, all of which implement the single-server algorithm for multidimensional navigation, and then to have a "master" server compute navigation states by passing requests onto the set of slave machines and combining the responses. From the outside, the collection of servers appears to act like a single server, but with far greater computational resources than would be possible on a single computational device. Indeed, the distinction between master and slave servers is arbitrary; a slave server can itself have slaves, thus creating a nested hierarchy of servers. Such nesting is useful when the number of slaves exceeds the fan-out capability of a single master server. An exemplary embodiment of such a system is illustrated in Figure 24. In the hierarchical arrangement 500, a master server 520 works with slave servers 530, 540. In the hierarchical arrangement shown, slave servers 530 are in turn master servers with respects to slave servers 540. The search and navigation results are made available to a user on a terminal 510 through a user interface in accordance with the present invention.

The collection of materials may be partitioned by splitting (arbitrarily or otherwise) the materials into disjoint subsets, where each subset is assigned to its own server. The subsets may be roughly equal in size, or they might vary in size to reflect the differing computational resources available to each server.

The algorithm for distributing the task of computing the information associated with a navigation state includes three steps. The steps of the algorithm are indicated in Figure 24. In the first step, the query, which is a request for a valid navigation state, is submitted to the master server 520, which forwards the query to each of the slave servers 530. If the servers are nested, the requests are forwarded through the hierarchy of servers 500 until they reach the leaf servers 540 in the hierarchy. In the second step, each slave server 530, 540 processes the query independently, based on the subset of the collection of materials that is in its partition. In the third step, the master server 520 combines the responses from the slave servers to produce a response for the original query. The master server 520 returns the response to the terminal 510.

The master server receives the original request and farms it out to the slave servers. Thus, in preferred embodiments, the only computation performed by the master server is to combine the results from the slave servers. Each slave server that receives a request computes the navigation state based on the subset of the collection assigned to it. The computation may involve any combination of conjunction, disjunction, and negation.

The master server, in contrast, only performs a combination step. The combination step involves producing a valid navigation state, including documents and corresponding refinement options, from the responses from the slave servers. Since the collection of materials has been partitioned into disjoint subsets, the documents identified by each of the slave servers can be combined efficiently as a disjoint union. Combining the various refinement options returned by each of the slave servers may require additional processing, as described below.

The slave servers all process the same query, but on different partitions of the collection of materials. They will generally return different sets of refinement options because a set of refinement options that is valid for one partition may be invalid for another. If the different sets are disjoint, and if the refinement options involve terms that do not themselves have refinement relationships, then the combination is a disjoint union.

Typically, there will be some overlap among the different sets of refinement options returned by each slave server. If the sets are not disjoint, duplicates can be eliminated in this combination step.

When there are refinement relationships among the terms that are refinement options returned by the slave servers, the combination algorithm computes, for every set of related terms, the least common ancestor or ancestors (LCA) of the terms, as defined by the partial order among the terms. One algorithm for combining the refinement options is outlined in Figure 25. In step 552, the master server receives and takes the union of all of the terms, $x_1, x_2, \ldots x_n$, returned as refinement options for the navigation state from the slave servers. In step 554, the master server computes the set of ancestors $A_1, A_2, \ldots$ An, for each of the terms, $x_1, x_2, \ldots$ xn, respectively. In step 556, the master server computes the intersection A of all of the sets of ancestors, $A_1, A_2, \ldots$ An. In step 558, the master server computes the set M of minimal terms in A. The set M, formed of the least common ancestors of the terms $x_1, x_2, \ldots x_n$, returned by the slave servers, is the set of refinement options corresponding to the result navigation state. This combination procedure is applied whether the refinement options are conjunctive, disjunctive, or negational.

In summary, the master server receives a request for a navigation state, forwards this request to each of the slave servers, combines their results with a union operation, and then computes, for every set of terms, the least common ancestor or ancestors of the set.

There are at least two ways to compute the LCA of the terms. One approach is to store all non-leaf terms on the master server. This strategy is reasonably memory efficient, since, in practice, most of the terms are leaves (minimal elements) in the partial order. A second approach is to include the ancestors when returning the terms that are refinements. This approach saves memory at the expense of increasing the size of the data being transferred. The latter overhead is reasonable, since, in practice, a term typically has very few ancestors.

The task of computing results for a free-text search query may also be distributed. In the arrangement described above, for example, the master can simply compute the union of the free-text search results returned by the slave servers. This approach applies to both single-term and multi-term search under both conjunctive and disjunctive query semantics. More complex approaches may be used to accommodate customized query semantics.

The search and navigation system of the present invention allows information providers to overlay a search and navigation system over any collection of documents. The knowledge base aspect and the search and navigation aspect of the invention can be performed independently by different providers, and information providers may outsource these functions to separate entities. Similarly, a generated knowledge base

may be imported by a search and navigation specialist. Information providers may also outsource this search and navigation requirement to a search and navigation system provider. A search and navigation system provider could charge customers a license fee for the system independent of the amount of its usage. Alternatively, a search and navigation system provider could charge customers on a per-click basis, a per-purchase basis if products are available via the system, or per-transaction generated from a click through the search and navigation system. A search and navigation system provider could also function as an aggregator -- compiling records from a number of sources, combining them into a global data set, and generating a search and navigation system to search and navigate the data set. The search and navigation system can be implemented as software provided on a disk, on a CD, in memory, etc., or provided electronically (such as over the Internet).

A search and navigation system in accordance with the present invention may also enhance user profiling capability and merchandising capability. The search and navigation system may maintain a profile of users based on the users' selections, including the particular paths selected to explore the collection of navigation states. Using the knowledge base, the system may also infer additional information regarding the users' preferences and interests by supplementing the selection information with information regarding related documents, attributes and terms in the knowledge base. That information may be used to market goods and services related to the documents of interest to the user.

The foregoing description has been directed to specific embodiments of the invention. The invention may be embodied in other specific forms without departing from the spirit and scope of the invention. The embodiments, figures, terms and examples used herein are intended by way of reference and illustration only and not by way of limitation. The scope of the invention is indicated by the appended claims and all changes that come within the meaning and scope of equivalency of the claims are intended to be embraced therein.

We claim: